Conociendo Arrow Functions

Felipe Nascimento

Actualizado en 04/10/2021

Una escuela necesita imprimir una lista de todos os funcionarios y el código del grupo y para realizar esa acción tenemos la siguiente clase:

```
class ModuloDeImpresion {
    constructor() {
        this._codigo = 10;
    }
    imprime(nombres) {
        nombres.forEach(function(nombre){
        console.log(`${this._codigo}: ${nombre}`);
     });
```

```
    }
}
```

El problema es que después de la prueba inicial:

```
const profesores = ['Elias', 'Yuri', 'Gabriel', 'Guilherme', 'Yan'];
const impresion = new ModuloDeImpresion();
impresion.imprime(profesores);
```

Está siendo mostrado en la consola el siguiente error:

Lo que vamos hacer es descubrir el por qué y cómo resolver ese error.

Entendiendo el error

El JavaScript nos está diciendo que `this._codigo` es `undefined`, o sea, no tiene ningún valor atribuido a él. Eso ocurre porque **el `this` dentro de una función tiene un comportamiento dinámico**, es decir, varía de acuerdo con el contexto de ejecución.

En nuestro caso, ese error ocurre porque, en el momento de ejecución del `forEach`, el contexto del `this._codigo` es el contexto de la función pasada para el `forEach` y no el de la clase `ModuloDeImpresion` que contiene el constructor con el código del grupo.

Contexto

Lo que queremos decir con **contexto** es que, durante la ejecución del `forEach` vamos a entrar a la propriedad donde está el **código del grupo**, pero no vamos a conseguir encontrar esa propriedad, porque en ningún lugar existe el `this._codigo`, dentro de la función que el `forEach` está ejecutando.

O sea, fuera del contexto del `forEach` conseguimos entrar a la propriedad `this._codigo`:

```
class ModuloDeImpresion {
    constructor() {
        this._codigo = 10;
    }
    imprime(nombres) {
        console.log(this._codigo);
        nombres.forEach(function(nombres){
        console.log(`${this._codigo}: ${nombre}`);
     });
   }
}
const impresion = new ModuloDeImpresion();
impresion.imprime(professores);
```

Como resultado obtenemos:

Después que instanciamos a la clase nuevamente, podemos percibir que conseguimos imprimir el código sin problemas, pero aún estamos con error. Lo que queremos es una manera de entrar el contexto de la clase `ModuloDeImpresion` en el momento en que ejecutamos el bucle `forEach`.

Arrow Function

La versión del ECMA Script 2015 del JavaScript, trajo una nueva forma más breve de trabajar con funciones llamada de **Arrow Functions**, por causa de la sintaxis que recuerda una flecha: `() =>`.

Pero la Arrow Function no es solo una manera menos verbosa de escribir una función, ella tiene una característica en particular que nos va auxiliar en nuestro problema: **el ámbito léxico**. ¿Pero cómo así el **ámbito léxico**?

Ámbito Léxico

Ambito léxico, significa que podemos entrar a la propiedad **código** dentro de nuestro `forEach`. El `this` no irá variar de acuerdo con el **contexto**. Ahora, como el contexto del `this._codigo` es el de la clase `MóduloDeImpresion` tenemos acceso a la propriedad código:

```
nombres.forEach((nombre) => {
    console.log(`${this._codigo}: ${nombre}`);
});
```

Ahora que sabemos cómo funciona el`this` y la arrow function, vamos aplicar en nuestro problema.

Resolviendo el problema

Ahora utilizando una arrow function en nuestra clase, tenemos acceso al constructor en el momento en que entramos en el bucle del`forEach`, porque el `this._codigo` va estar siempre en el contexto de la clase `ModuloDeImpresion`.

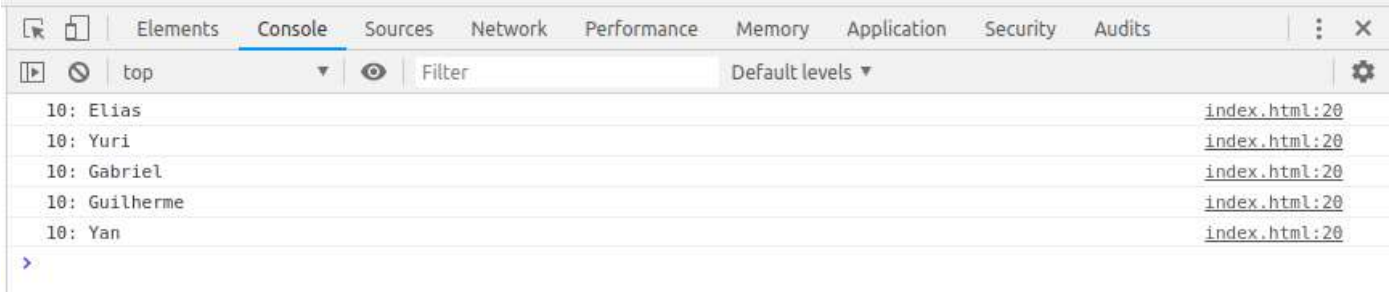
Después de aplicar la técnica de la arrow function, nuestro código quedó así:

```
class ModuloDeImpresion {
    constructor() {
        this._codigo = 10;
    }
    imprime(nombres) {
        nombres.forEach(nombre => {
        console.log(`${this._codigo}: ${nombre}`);
    });
  }
}
```

cuando instanciamos un nuevo`ModuloDeImpresion`:

```
const profesores = ['Elias', 'Yuri','Gabriel', 'Guilherme','Yan'];
const impresion = new ModuloDeImpresion();
impresion.imprime(profesores);
```

Tenemos como resultado en la consola:



Muy bien, ¡conseguimos resolver nuestro problema con relación al contexto del `this`! Ahora ya tenemos el código del grupo y el nombre de los profesores.

Otra solución

Además de arrow functions, otra manera que podemos solucionar ese problema es a través del método **bind**, que va fijar un valor para el `this`, así el no irá variar de acuerdo con el contexto en que está insertado

```
class ModuloDeImpresion {
    constructor() {
        this._codigo = 10;
    }
    imprime(nombres) {
        nombres.forEach(function(nombre) {
        console.log(`${this._codigo}: ${nombre}`);
     }.bind(this));
   }
}
```

Como podemos ver, el `bind` va pasar un contexto para una función que no es de ella, o sea, el `this._codigo` dentro del `forEach` va tener el contexto de la clase `ModuloDeImpresion`.

Para saber más

Las **Arrow functions** tiene otras propiedades interesantes como:

- **Retorno:** No necesitamos declarar explícitamente el retorno cuando tenemos un bloque apenas de código, la última expresión realizada va ser el retorno de la arrow function.

```
const suma = (numero1, numero2) => numero1 + numero2;
suma(3,7) // 10
```

- **Nombre:** Las Arrows functions ganan nombre de la variable atribuida cuando de la criación

```
const arrow = () => {};
arrow.name; // arrow
```

- **Constructor:** No es posible crear constructores con Arrow function

```
const Construtor = () => {};
new Construtor(); // Constructor is not a constructor
```

Si quedaste interesado en cómo el Javascript funciona y como tú puedes utilizarlo mejor, aquí en Alura tenemos una **formación en desarrollo Javascript**. En ella, tú veras como programar en Javascript, utilizar expresiones regulares, entre otras muchas cosas.

Felipe Nascimento

Desarrollador e instructor en Alura con foco en JavaScript.

Puedes leer también:

- Creando un autocomplete con JavaScript
- Cambiando CSS con JavaScript
- Crear una máscara de Teléfono con Javascript

ARTÍCULOS DE TECNOLOGÍA > FRONT END

¡QUIERO EMPEZAR A ESTUDIAR!

Paga en moneda local en los siguientes países

ANUAL

US$79,90

un solo pago de US$79,90

- 218 cursos
- Videos y actividades 100% en Español
- Certificado de participación
- Estudia las 24 horas, los 7 días de la semana
- Foro y comunidad exclusiva para resolver tus dudas
- Acceso a todo el contenido de la plataforma por 12 meses

¡QUIERO EMPEZAR A ESTUDIAR!

Lógica de Programación | Java

Cursos de Front End

HTML y CSS | JavaScript | React

Cursos de Data Science

Data Science | Machine Learning | Excel | Base de Datos | Data Visualization | Estadística

Cursos de DevOps

Docker | Linux

Cursos de Innovación y Gestión

Productividad y Calidad de Vida | Transformación Ágil | Marketing Analytics | Liderazgo y Gestión de Equipos | Startups y Emprendimiento